Rapport Pimp' My Fridge

Pons Emilien

Deltour Hugo

Exia A2

Introduction:

Dans le cadre d'un projet d'apprentissage au sein de l'école du CESI, ce projet englobe quatre notions :

- Mesures Physiques
- Automatique
- Thermodynamique
- Java

Nous disposons d'un mini-frigo USB qui refroidit notre boisson préférée, mais la température extérieure influence la température de notre boisson, et la boisson est souvent trop fraîche, ne permettant pas d'en apprécier les arômes. On y retrouve de la **condensation**. De plus, il est nécessaire que nous puissions **modifier la température de consigne** à tout moment, de **suivre la température intérieure**, **extérieure** et **l'humidité**, mais aussi **d'alerter en cas de condensation** ou d'augmentation anormale de la température (Porte ouverte).

Partie 1 – Conceive :

La première partie de notre projet, en plus de l'étude des différents phénomènes physiques et thermodynamiques, a été de mettre en place une organisation interne.

Ces documents sont disponibles en consultation sur le Git fournit.

- **Cahier des charges fonctionnel :**

Ce document est la liaison entre le client et le concepteur, il définit par des fonctions (principales et secondaires) la demande, les critères d'appréciations, les niveaux, et d'éventuelles flexibilités.

- **PBS (Project Breakdown Structure) :**

Document qui va lister tout les livrables, à savoir : un poster, un Git, les documents de gestion de projets, un Trello, de la modélisation, et la partie applicative

- **WBS (Work Breakdown Structure) :**

A partir des éléments listés dans le PBS, nous pouvons décomposer. Par exemple, pour la gestion de projet, nous avons le PBS, le WBS, l'OBS, le Pert et le Gantt.

- **OBS (Organisation Breakdown Structure) :**

En vue de la décomposition du WBS, il faut maintenant attribuer chaque personne les tâches. On peut s'appuyer sur la matrice RACI pour faire cette répartition.

- **GANTT :**

C'est un calendrier auquel nous pouvons ajouter les tâches, et leur attribuer des ressources (développeurs) en s'inspirant de l'OBS. C'est le principal de la planification

- **PERT :**

En prenant compte des dates définis dans le GANTT, on peut calculer plusieurs chemins d'arriver à notre objectif final. En observant ces différents chemins, on peut trouver le chemin critique.

- **Trello :**

C'est l'interface graphique que le développeur consulte régulièrement, et met à jour. On y retrouve des checks, et la liste des tâches. A chaque fois qu'une tâche est faite, elle doit être mise comme check, puis déplacé dans la bonne colonne.

- **Rapport :**

Ecrit tout au long, qui retrace les principes utilisés.

Partie 2 – Conceive :

La seconde partie de notre projet concerne la modélisation.

- **UML**
    - Diagramme activité :
        - Reprends le fonctionnement de la récupération de données vers l'affichage, montre le fonctionnement de notre système d'un point A à un point B.
    - Diagramme cas utilisation :
        - Montre les interactions entre le système et l'utilisateur (Externe). Il peut donc modifier la température, ou consulter les données des relevés thermodynamique.
    - Diagramme de séquence :
        - Montre le fonctionnement Java, depuis la classe CAD quand on se connecte à la carte arduino, vers le modèle qui joue le rôle d'observable avec le contrôleur (Observable), jusqu'à l'affichage de la vue. C'est la suite des exécutions des méthodes dans un exemple définit.
    - Diagramme de classe :
        - Reprends l'intégralité des attributs, constructeurs et méthodes de notre code Java, prenant en compte les liaisons entre chaque classe. On y retrouve des héritages, des implémentations, de la composition, et de l'agrégation.
    - Diagramme de composants :
        - Il représente les interactions entre les différents composants de notre système. Malheureusement, celui-ci n'a pas été instancié par manque de temps.
- **Schéma Système :**
  Ce schéma montre comment marche notre système entre le bloc comparateur (Le cercle) qui va calculer entre notre température intérieure, et la température consigne, et en fonction de ce résultat, va décider si on envois ou non du courant dans le module à effet pelletier

(Commande), ce module à effet Pelletier va refroidir en conséquence notre système (Mini-Frigo USB). C'est un schéma simplifié en vue que de nombreuses choses se passent derrière, et il ne prend pas en compte toute la partie de l'affichage utilisateur.

- **IHM :**

C'est la modélisation de notre future vue (l'interface graphique avec l'utilisateur), où nous y posons les différents composants de manière graphique. La modélisation sera donc à respecter lors de la réalisation.

- **Schéma Electrique :**

Il représente tous les branchements électriques de notre système. On y retrouve deux parties.

Une partie « récolte de données », qui va comprendre une thermistance (Température Extérieure), une sonde DHT, (Température intérieure + Humidité). La récolte de la thermistance repose sous le montage du pont diviseur de tension.

Une partie « Contrôle » qui va contrôler un ventilateur et un module à effet Peltier, en passant par des transistors.

Le tout est contrôlé sur une carte Arduino.

Les détails des calculs suivront plus tard.

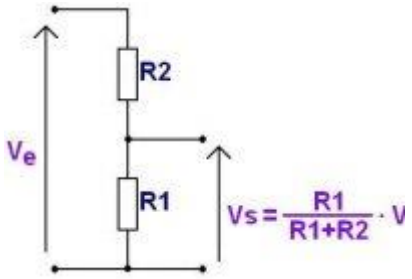
Partie 3 – Implement :

Arduino :

Ce programme va récupérer les valeurs de la thermistance et de la sonde DHT.

- **Thermistance :**

On récupère la tension de la thermistance, puis à l'aide de la formule du pont diviseur de tension




```
En posant en équation, on récupère la valeur de la résistance, puis à
l'aide de Steinhardt HArt :
1/T = A + B*ln R + C*(ln R)^3
Où A, B et C sont des coefficients calculés dans le Workshop.
```

Nous récupérons la valeur de la température (T) étant la température.

- **Sonde DHT :**

Il suffit de lancer les méthodes de la bibliothèques DHT, pour récupérer les deux valeurs.

- **Ecriture :**
  De temps en tant, la carte va recevoir la température de consigne, il faut donc dans le programme (loop arduino), vérifier que la température consigne est inférieure à la température interne (récupéré par la DHT). Si c'est le cas, on alimente le PIN 8, qui va alimenter le transistor, et donc déclencher l'alimentation du ventilateur + module à effet Peltier.

Java :

On retrouve la structure MVC (Modèle Vue Contrôleur), et une classe CAD.

- CAD :

Se connecte à la carte arduino (via un port USB), récolte les données que récolte la sonde, et la thermistance.

Il va envoyer les données vers le modèle, puis va lire la valeur de consigne, l'envoyant vers la carte, et ainsi de suite… En boucle.

- Modèle :

C'est la database, ses données vont-êtres mises à jour par la CAD toutes les 5 secondes. Mais c'est aussi un observable, c'est-à-dire qu'à chaque fois que ses données vont-être MAJ, les observer vont aussi être MAJ. C'est le cas du contrôleur.

- Contrôleur :

C'est la couche de contrôle, il va recevoir les données du Modèle à chaque fois que la CAD en envois, en vue que c'est un observer.

Lorsqu'il met à jour ses données, il va aussi mettre à jour la vue, en fonction des dernières valeurs reçues. Il va calculer le point de rosée, et un delta de température, pour déterminer si la porte est ouverte. Il va aussi récupérer la température de consigne, lorsque l'utilisateur effectue une action sur la vue.

- Vue :

C'est la couche d'affichage de notre programme, la couche d'interaction avec l'utilisateur. On retrouve ce qu'on a mis dans notre IHM précédemment crée.

Partie 4 – Operate :

C'est la couche maintenance. Nous avons conçus une JavaDoc pour permettre à l'utilisateur de connaître les différentes fonctionnalités, et de nombreux commentaires sur le code permette d'avoir de la compréhension.